initTransactions

A. Producer and transaction coordinator

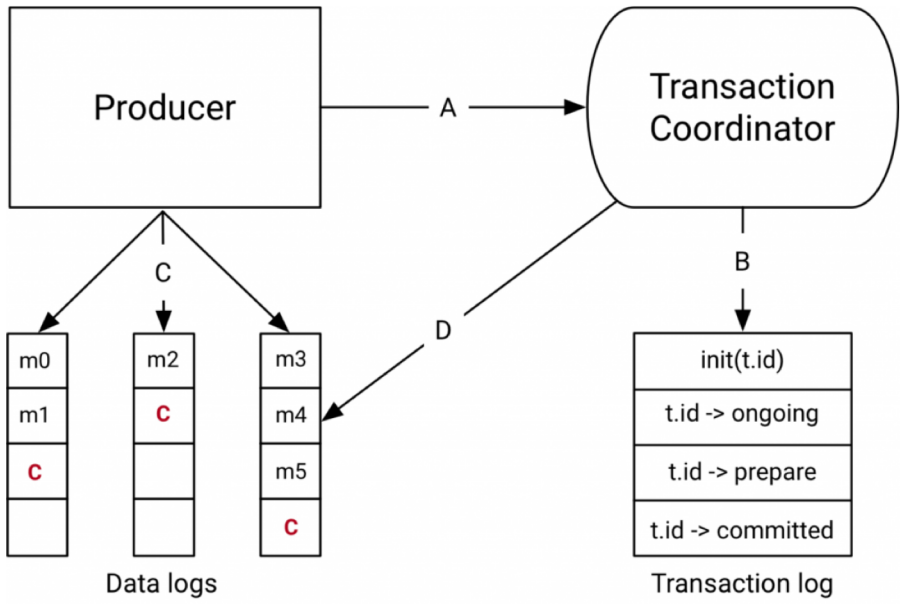
Transaction Started

Producer registers a transactional.id with the coordinator.

Coordinator closes any pending transactions with that transactional.id and bumps the epoch to fence out zombies.

If partition for the first time in a transaction, the partition is registered with the coordinator first.

Any producers with same transactional.id and an older epoch are considered zombies and are fenced off, i.e. future transactional writes



When a transaction is started by the listener container, the transactional.id is now the transactionIdPrefix appended with <group.id>.<topic>.<partition>.

As the transaction progresses, the producer sends the requests above to update the state of the transaction on the coordinator. The transaction coordinator keeps the state of each transaction it owns in memory, and also writes that state to the transaction log.

The transaction coordinator is the only component to read and write from the transaction log.

B. the coordinator and transaction log interaction

Once the markers are written, the transaction coordinator marks the transaction as “complete” and the producer can start the next transaction.

phase 2, where it writes *transaction commit markers* to the topic-partitions which are part of the transaction.

Phase 1, the coordinator updates its internal state to “prepare_commit” and updates this state in the transaction log.

producer initiates a commit (or an abort), the coordinator begins the two phase commit protocol.

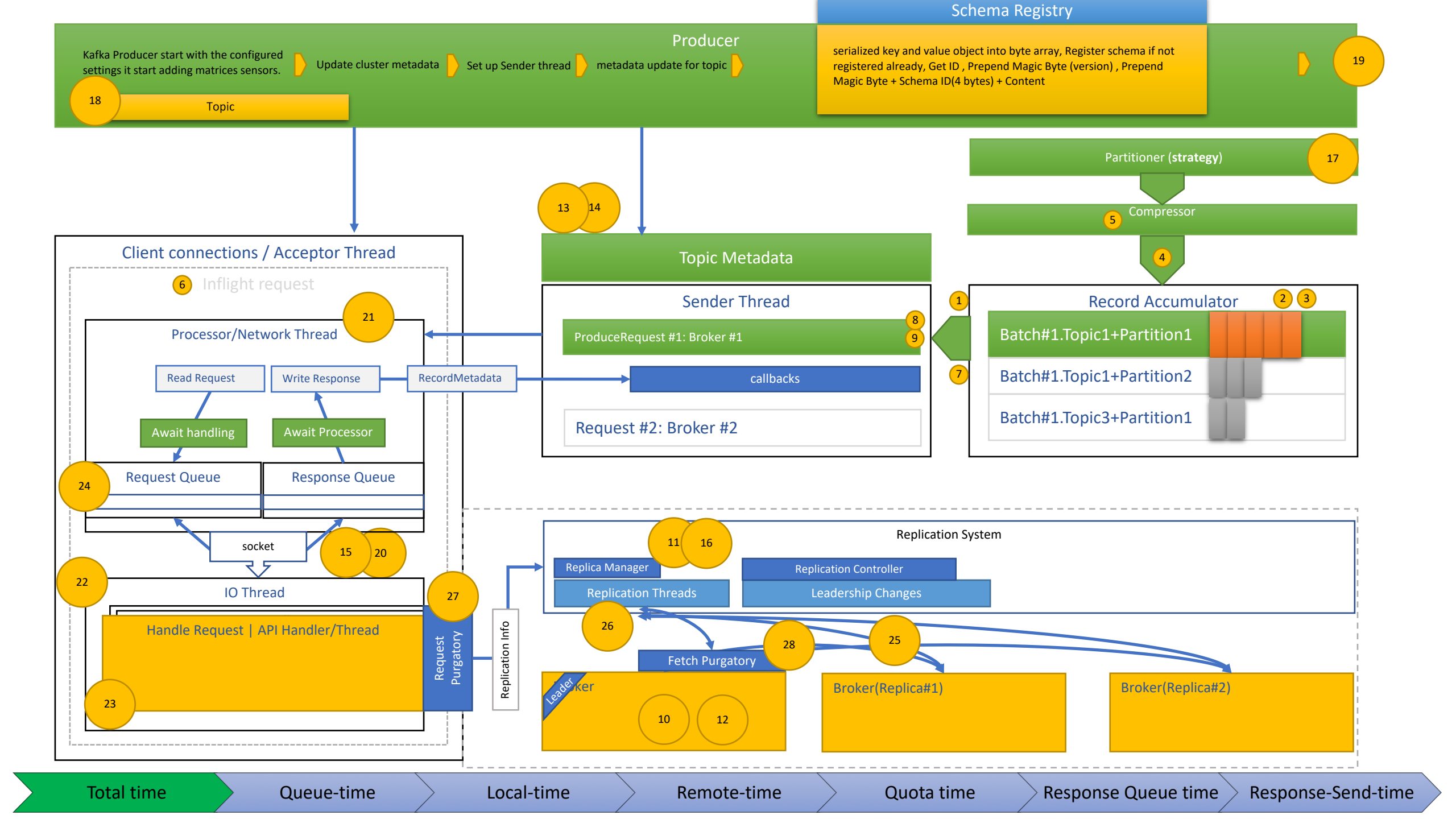
D. coordinator to topic-partition interaction

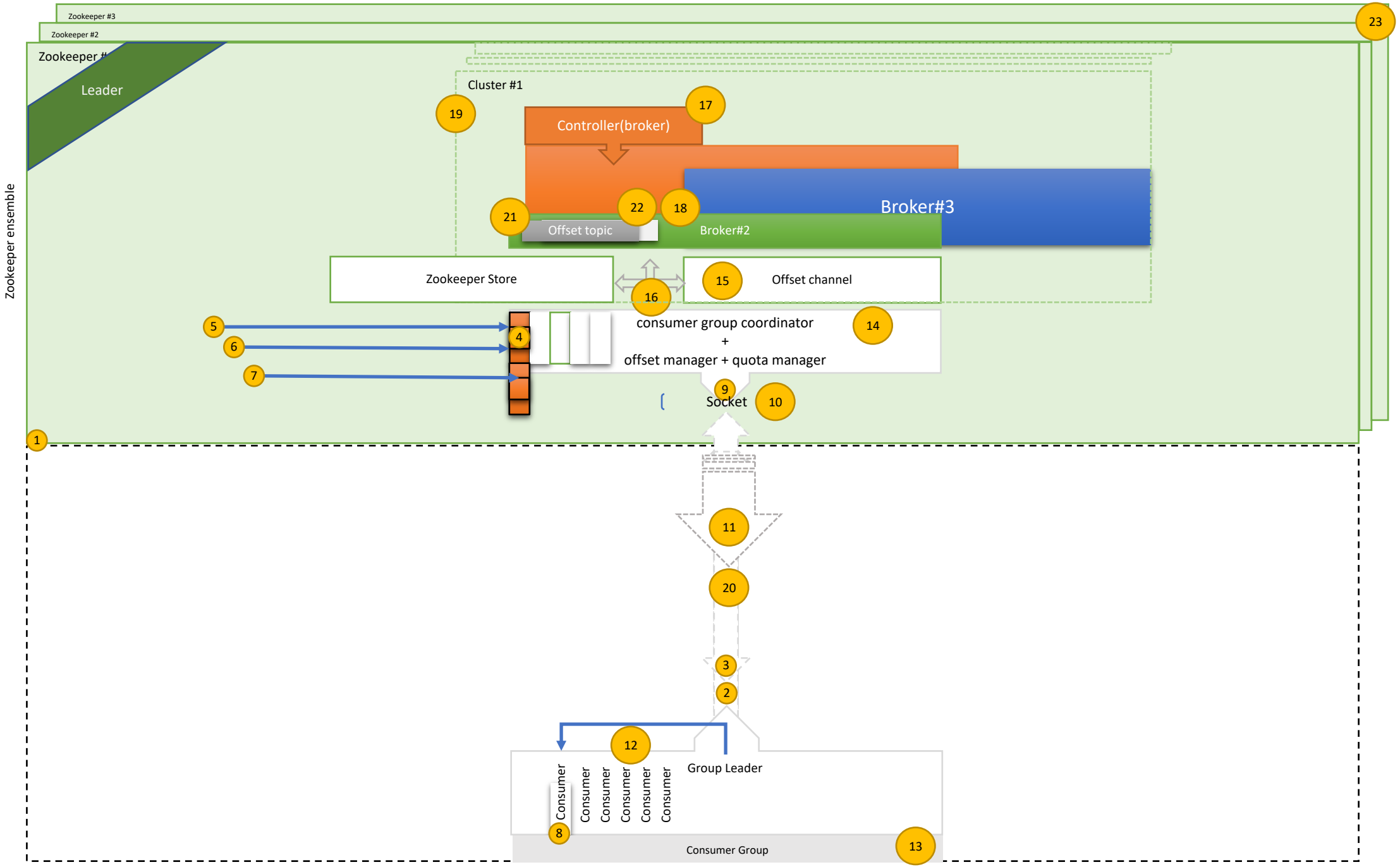
Commit /Abort Transaction

After registering new partitions in a transaction with the coordinator, the producer sends data to the actual partitions as normal.

C. Producer writing data to target topic-partitions

Produce


Schema Registry
Producer
Kafka Producer start with the configured settings it start adding matrices sensors.
Update cluster metadata
Set up Sender thread
metadata update for topic
Topic
18
serialized key and value object into byte array, Register schema if not registered already, Get ID , Prepend Magic Byte (version) , Prepend Magic Byte + Schema ID(4 bytes) + Content
19
Partitioner (strategy)
17
Compressor
5
4
Topic Metadata
13
14
Client connections / Acceptor Thread
6
Inflight request
Processor/Network Thread
21
Read Request
Write Response
RecordMetadata
Await handling
Await Processor
Request Queue
Response Queue
24
socket
15
20
22
IO Thread
Handle Request | API Handler/Thread
23
27
Request Purgatory
Replication Info
Sender Thread
ProduceRequest #1: Broker #1
8
9
callbacks
Request #2: Broker #2
1
7
Record Accumulator
2
3
Batch#1.Topic1+Partition1
Batch#1.Topic1+Partition2
Batch#1.Topic3+Partition1
Replication System
11
16
Replica Manager
Replication Threads
Replication Controller
Leadership Changes
26
28
25
Fetch Purgatory
Leader
10
12
Broker(Replica#1)
Broker(Replica#2)
Total time
Queue-time
Local-time
Remote-time
Quota time
Response Queue time
Response-Send-time


Zookeeper ensemble
Zookeeper #3
Zookeeper #2
Zookeeper #
Leader
Cluster #1
Controller(broker)
Broker#3
Broker#2
Offset topic
Zookeeper Store
Offset channel
consumer group coordinator
+
offset manager + quota manager
Socket
Group Leader
Consumer
Consumer
Consumer
Consumer
Consumer
Consumer
Consumer Group
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

Log Manager
Loaded(default: /tmp/kafka-logs)

num.recovery.threads.per.data.dir dynamic configuration (default: 1) for the number of threads per log data directory for log recovery at startup and flushing at shutdown.

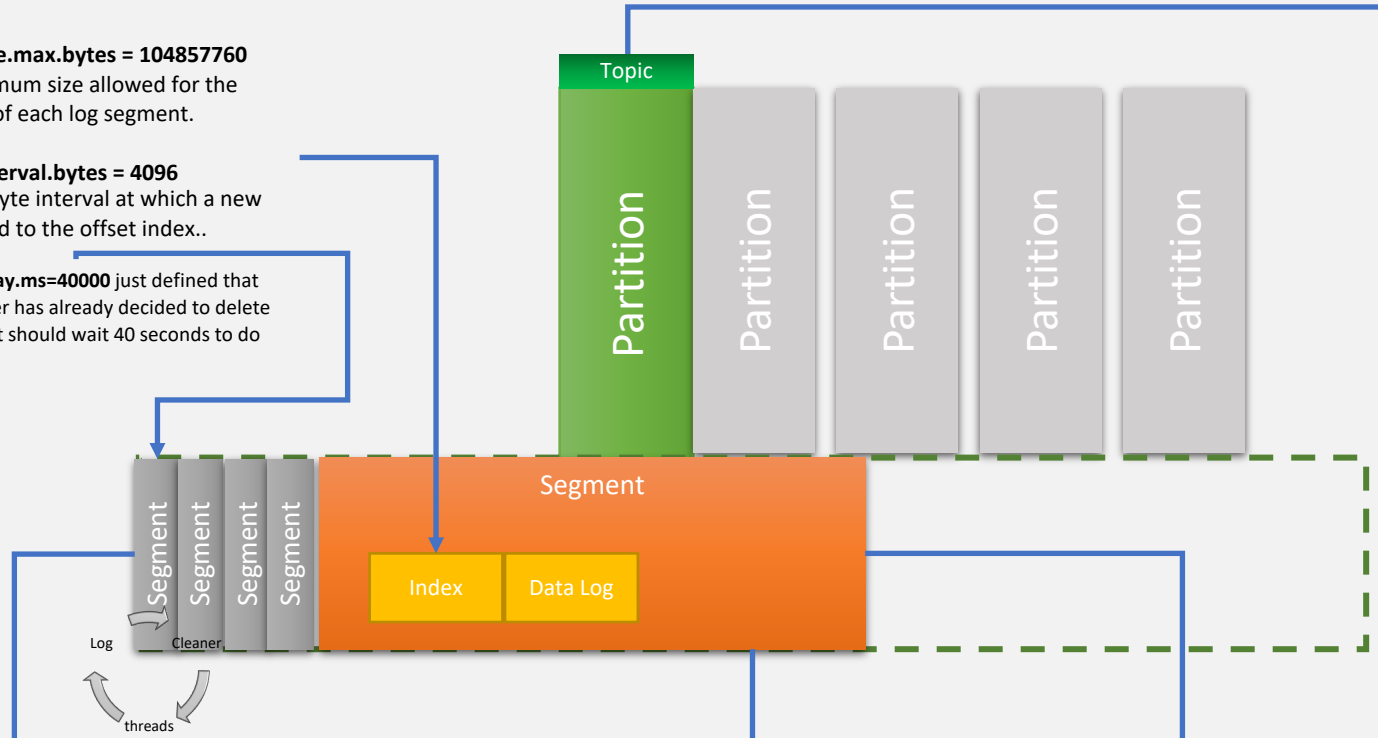
Thread#1: /tmp/kafka-logs)

Thread#2: /tmp/kafka8182-logs)

log.index.size.max.bytes = 104857760
set the maximum size allowed for the offset index of each log segment.

log.index.interval.bytes = 4096
defines the byte interval at which a new entry is added to the offset index..

file.delete.delay.ms=40000 just defined that after the broker has already decided to delete that segment it should wait 40 seconds to do that



log.retention.hours/minutes/ms = 168
define the time a message is stored on a topic before it discards old log segments to free up space.

log.retention.bytes = -1
maximum number of log bytes per partitions that is retained before its deleted. Can be set per topic basis, when either log time or size limit is reached – the segments are deleted.

log.retention.check.interval.ms = 30000

time interval at which the logs are checked for deletion to meet retention policies, default is 5 minutes.

log.cleaner.enable = false
for the log compaction to be enabled, make it true.

log.cleaner.threads = 1
set the number of threads that need to be working to clean logs for compaction

log.cleaner.backoff.ms=15000
interval at which the logs will check whether any logs need cleaning.

log.cleanup.policy=delete
if value is set to delete, log segments will be deleted periodically when it reaches its time threshold or size limit. If the log compaction is set, log compaction will be used to clean-up obsolete records, can be set for topic basis.

log.cleaner.min.compaction.lag.ms
This can be used to prevent messages newer than a minimum message age from being subject to compaction.

log.segment.ms (time - default a week)
ask how frequently log compaction need to happen

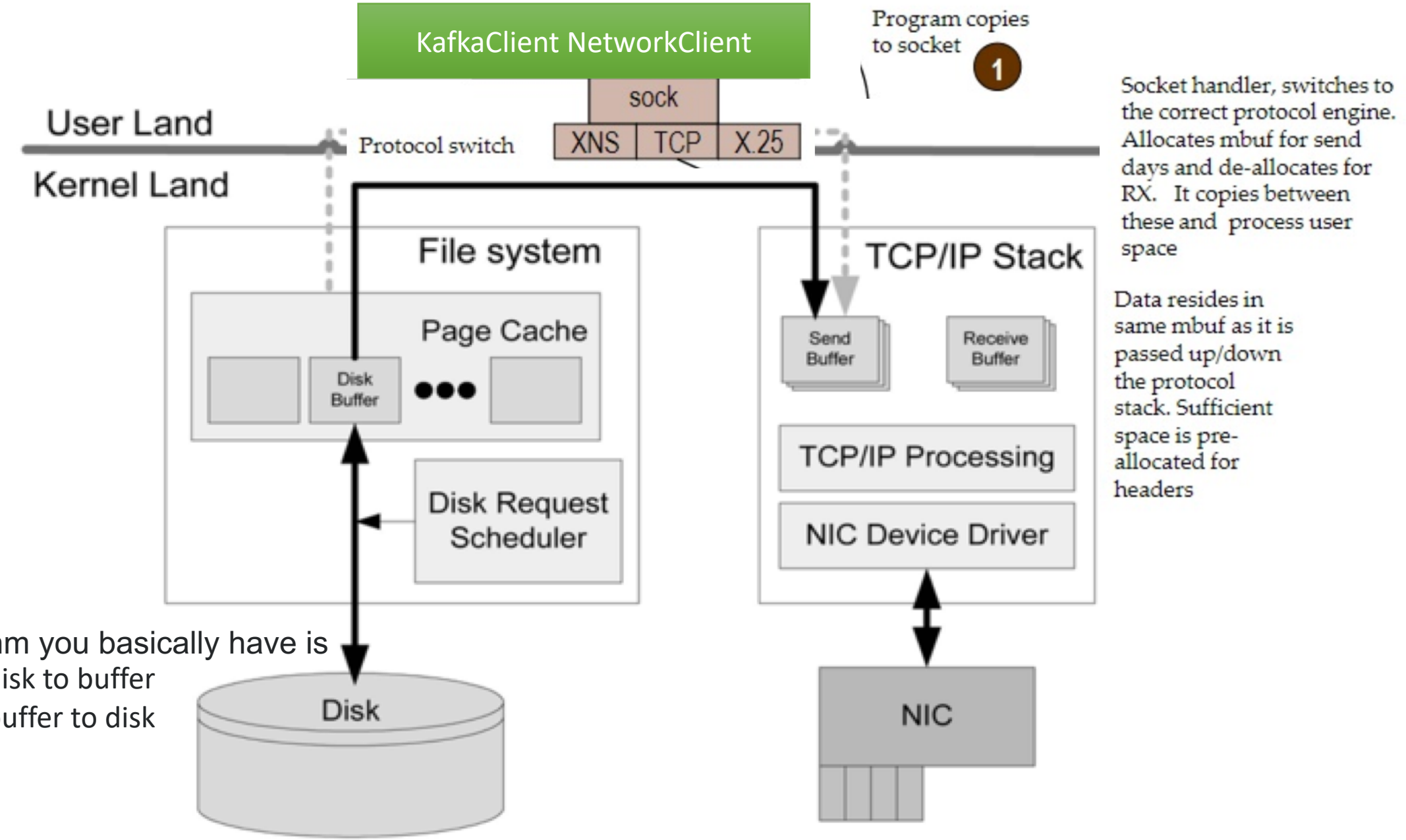
log.segments.bytes
maximum segment size in bytes, once segment reaches that size, a new segment file is created, a topic is stored as bunch of segment files in a directory. Default value is 1 GB

log.roll.{hours,ms}= 168
sets the time period after which a new segment file is created, even if it has not reached the size limit. This settings can be set for topic basis, default is 7 days.

log.flush.interval.messages = Long.MaxValue
Sets the number of messages that are kept in memory till they are flushed to the disk.

File system

long.flush.interval.ms = Long.MaxValue
sets the time interval at which the messages are flushed to the disk



The algorithm you basically have is

- copy from disk to buffer
- copy from buffer to disk

Log Manager
Loaded(default: /tmp/kafka-logs)

num.recovery.threads.per.data.dir dynamic configuration (default: 1) for the number of threads per log data directory for log recovery at startup and flushing at shutdown.

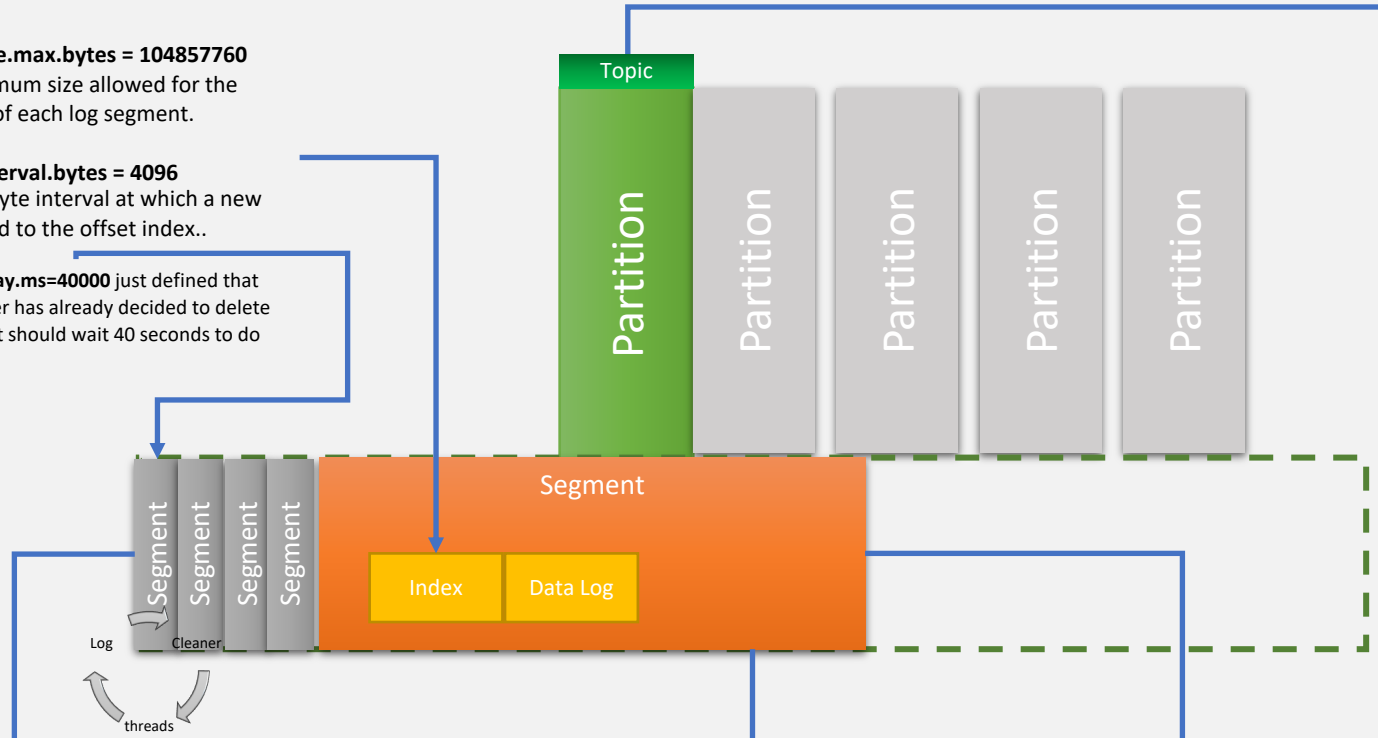
Thread#1: /tmp/kafka-logs)

Thread#2: /tmp/kafka8182-logs)

log.index.size.max.bytes = 104857760
set the maximum size allowed for the offset index of each log segment.

log.index.interval.bytes = 4096
defines the byte interval at which a new entry is added to the offset index..

file.delete.delay.ms=40000 just defined that after the broker has already decided to delete that segment it should wait 40 seconds to do that



log.retention.hours/minutes/ms = 168
define the time a message is stored on a topic before it discards old log segments to free up space.

log.retention.bytes = -1
maximum number of log bytes per partitions that is retained before its deleted. Can be set per topic basis, when either log time or size limit is reached – the segments are deleted.

log.retention.check.interval.ms = 30000

time interval at which the logs are checked for deletion to meet retention policies, default is 5 minutes.

log.cleaner.enable = false
for the log compaction to be enabled, make it true.

log.cleaner.threads = 1
set the number of threads that need to be working to clean logs for compaction

log.cleaner.backoff.ms=15000
interval at which the logs will check whether any logs need cleaning.

log.cleanup.policy=delete
if value is set to delete, log segments will be deleted periodically when it reaches its time threshold or size limit. If the log compaction is set, log compaction will be used to clean-up obsolete records, can be set for topic basis.

log.cleaner.min.compaction.lag.ms
This can be used to prevent messages newer than a minimum message age from being subject to compaction.

log.segment.ms (time - default a week)
ask how frequently log compaction need to happen

log.segments.bytes
maximum segment size in bytes, once segment reaches that size, a new segment file is created, a topic is stored as bunch of segment files in a directory. Default value is 1 GB

log.roll.{hours,ms}= 168
sets the time period after which a new segment file is created, even if it has not reached the size limit. This settings can be set for topic basis, default is 7 days.

log.flush.interval.messages = Long.MaxValue
Sets the number of messages that are kept in memory till they are flushed to the disk.

File system

long.flush.interval.ms = Long.MaxValue
sets the time interval at which the messages are flushed to the disk

Kafka Broker Internals